ELSEVIER Discrete Applied Mathematics 49 (1994) 51–59

DISCRETE APPLIED MATHEMATICS

Solving the uncapacited plant location problem on trees

Alain Billionnet[a], Marie-Christine Costa[b]

[a]CEDRIC, Institut d'Informatique d'Entreprise, 18 allée Jean Rostand, 91025 Evry, France
[b]CEDRIC, Conservatoire National des Arts et Métiers, 292 rue Saint-Martin, 75141 Paris, France

Received 18 February 1991; revised 7 February 1992

Abstract

We propose an $O(n^2)$ algorithm for the uncapacited plant location problem on a given tree network of n vertices.

1. Introduction

Let $T = (V,E)$ be a tree with vertex set $V = \{1, \dots, n\}$ and edge set E. Each edge e of E has a positive length $l(e)$. The distance $d_{r,s}$ between two points r and s on T is defined to be the length of the shortest path between r and s, the length of a path being the sum of the lengths of its edges; in particular, $d_{r,r} = 0$. Clients are located at vertices of the tree and the facility locations belong to the vertex set. Each facility can serve all clients and the cost of establishing a facility at r is given by $f_r > 0$. Demand for the commodity from the client located at s is w_s and there is a transportation cost associated with transporting goods from a facility to a client which is linear with respect to the distance traveled. The uncapacited plant location problem (u.p.l.p.) is to minimize the sum of the costs of establishing the facilities and transporting the commodities. This problem can be formulated as the search for a subset R of $\{1, \dots, n\}$ which minimizes the set function

$$z(R) = \sum_{r \in R} f_r + \sum_{s=1,n} w_s \cdot \min_{r \in R} d_{r,s}.$$

R is the set of vertices where a facility is open and the demand of the client located at s is supplied by the nearest open facility [3]. This problem with $d_{r,s}$ replaced by arbitrary transportation cost $h_{r,s}$, is the standard formulation of the u.p.l.p. The general plant location problem is NP-hard. Kolen [2] has developed a polynomial-time algorithm that finds the solution in $O(n^3)$ time for the problem defined on a tree. His algorithm uses an algorithm due to Hoffman, Kolen and Sakarovitch [1] which can solve polynomially 0-1 linear programming problems when the constraints matrix satisfies certain properties. We adopt here a quite different approach specially adapted to this problem and we propose an $O(n^2)$ algorithm.

SSDI 0166-218X(92)00130-C

2. Definitions and notations

Transform the tree $T = (V, E)$ into a rooted tree $T = (V, A)$ by arbitrarily choosing, among the vertices of T, the vertex 1 as the root. (See Fig. 1(a) and (b)). For all $j \in \{1, \dots, n\}$ let V_j be the set of vertices composed of j and its descendants in $T = (V, A)$.

With each $R \subset \{1, \dots, n\}$ is associated one or several feasible solutions of the u.p.l.p. In the sequel we only consider feasible solutions where each client is supplied by the open facility located at the nearest vertex with the *smallest number*. This implies that a client located at a vertex k which is a successor of j in $T = (V, A)$, is supplied either by a facility in V_k or by a facility out of V_k but supplying j. Every feasible solution (and in particular an optimal solution) thus defines a forest F on the set of vertices V: the roots of the rooted trees of F are the vertices belonging to R (i.e., the vertices where a facility is open), and the arcs of F are the edges of E on the paths joining r and s, directed from r to s, for all (r, s) such that client s is supplied by a facility in r. Conversely, let $\mathscr{F}$ be the set of forests (V, A_F) such that $(s, t) \in A_F$ implies $[s, t] \in E$; every forest F of $\mathscr{F}$ defines a feasible solution of the u.p.l.p.: a client located at a vertex s is supplied by the facility open at the root of the tree of F containing s. Finally the u.p.l.p. can be formulated as the search for an optimal forest among the forests of $\mathscr{F}$. Note that we consider several directions of the edges of E: one direction creating an order among the vertices of V (the rooted tree $T = (V, A)$) and the others corresponding to the forests of $\mathscr{F}$, i.e., to feasible solutions. Fig. 1 illustrates the above concepts for a particular example.

Given two vertices i and j of V, denote by μ_{ij} the directed path from i to j obtained from the (undirected) path between i and j in $T = (V, E)$. Let $\mathscr{V}(\mu_{ij})$ be the set of its vertices and $\mathscr{A}(\mu_{ij})$ the set of its arcs. Given a forest F and a vertex s of F, denote by $r^F(s)$ the root of the rooted tree of F containing s.

For all $(i, j) \in \{1, \dots, n\}^2$, denote by $\mathscr{F}(i, j)$ the set of forests, subgraphs of forests of $\mathscr{F}$, satisfying:

(i) the set of vertices is $V_{ij} = V_j \cup \mathscr{V}(\mu_{ij})$,
(ii) the set of arcs includes the arcs of μ_{ij},
(iii) i is a root.

Some examples are given in Fig. 2.

Let F be a forest of $\mathscr{F}(i, j)$ (note that for each vertex s of F, $r^F(s) \in V_j \cup \{i\}$). By definition, the cost of F will be

$$\gamma(F) = \sum_{r \mid r \in V_j \text{ and } r \text{ is a root}} f_r + \sum_{s \mid s \in V_j} w_s d_{r^F(s), s}.$$

Every forest F of $\mathscr{F}(i, j)$ corresponds to a feasible solution of the u.p.l.p. restricted to the subgraph of $T = (V, E)$ generated by the set of vertices V_{ij}, including a facility in i and such that the clients in j are supplied by the facility in i (all clients at a vertex of $\mathscr{V}(\mu_{ij})$ are thus served by the facility in i). Clients located at a vertex of V_j can be supplied by a facility located at i even if i does not belong to V_j. Note that the cost of F is the value of such a feasible solution, where the demands of clients located out of V_j are not considered and where the cost of establishing a facility at i is not taken in account if $i \notin V_j$.

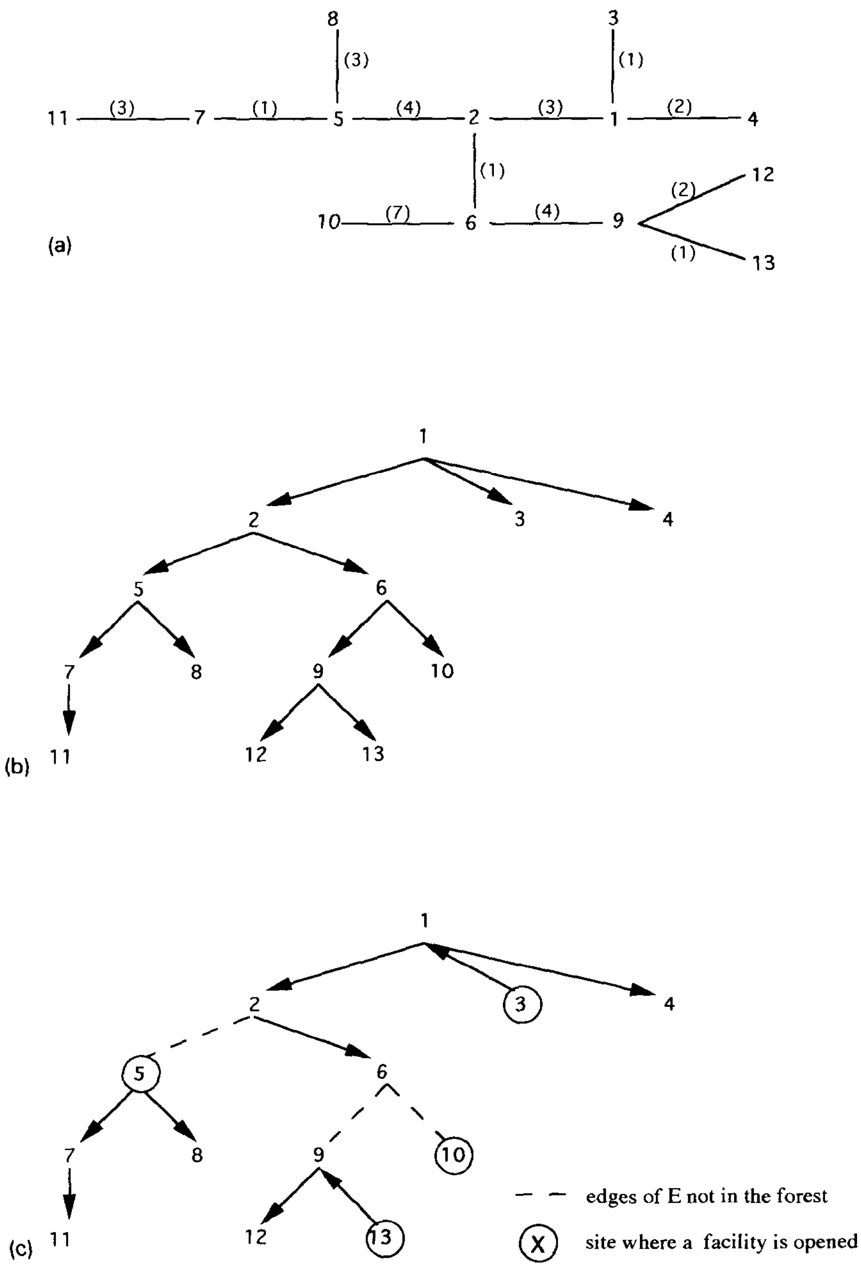

Fig. 1. (a) A tree network $T=(V,E)$. ((x): length of an edge, $d_{3,6}=1+3+1=5=$ distance between vertices 3 and 6.) (b) The rooted tree $T=(V,A)$. (c) F, the forest of $\mathscr{F}$ defined by $R=\{3,5,10,13\}$. (A client is supplied by the facility at the nearest vertex with the smallest number.)

Given two directed graphs $G' = (V', A')$ and $G'' = (V'', A'')$ we denote by $G' \cup G''$ the graph $(V' \cup V'', A' \cup A'')$. Let $\Gamma(j) = \{k: (j, k) \in A\}$, i.e., $\Gamma(j)$ is the set of successors of vertex j in the tree $T = (V, A)$.

3. Main results

We propose a method for computing, for all $(i, j) \in \{1, \ldots, n\}^2$, the cost of an optimal forest of $\mathscr{F}(i, j)$. Computations are carried out progressively using the fact that an optimal forest of $\mathscr{F}(i, j)$ can be determined from the optimal forests of $\mathscr{F}(r, k)$ for all $k \in \Gamma(j)$ and for all $r \in \{1, \ldots, n\}$. In this way we finally obtain the optimal cost of a forest of $\mathscr{F}$.

Lemma 1. *For all* $(i, j) \in \{1, \ldots, n\}^2$,

(i) $F \in \mathscr{F}(i, j) \Rightarrow \forall k \in \Gamma(j), \exists F_k \in \mathscr{F}(r^F(k), k)$ *such that* $F = \bigcup_{k \in \Gamma(j)} F_k \cup \mu_{ij}$,

(ii) $\forall k \in \Gamma(j), \forall r_k \in V_k \cup \{i\}, \forall F_k \in \mathscr{F}(r_k, k), F = \bigcup_{k \in \Gamma(j)} F_k \cup \mu_{ij} \Rightarrow F \in \mathscr{F}(i, j)$.

Proof. The proof is a direct consequence of the following:

– the definition of the set of forests $\mathscr{F}(i, j)$ for all $(i, j) \in \{1, \ldots, n\}^2$ (and therefore of $\mathscr{F}(r^F(k), k)$ and $\mathscr{F}(r_k, k)$);

– the definition of the union of two graphs;

– the tree structure of T which implies there is no edge between two vertices belonging respectively to V_k and V_l for all (k, l) such that $k \in \Gamma(j)$, $l \in \Gamma(j)$ and $k \neq l$;

– the facts that $r^F(k) = i$ if (j, k) is an arc of forest F (in this case μ_{ik} and μ_{ij} are directed paths of F and, of course, $\mu_{ik} = \mu_{ij} \cup (j, k)$) and that $r^F(k) \in V_k$ if (j, k) is not an arc of F (in this case $\mathscr{V}(\mu_{r^F(k), k}) \subset V_k$). $\square$

Lemma 1 is illustrated by Fig. 2.

Lemma 2. *Let* F be *a forest of* $\mathscr{F}(i, j)$ *and for all* $k \in \Gamma(j)$ *let* r_k *be a vertex of* $V_k \cup \{i\}$; *let* F_k *be a forest of* $\mathscr{F}(r_k, k)$. *If*

$$F = \bigcup_{k \in \Gamma(j)} F_k \cup \mu_{ij}$$

then

$$\gamma(F) = \sum_{k \in \Gamma(j)} \gamma(F_k) + w_j d_{i,j} \quad \text{for } j \neq i,$$

and

$$\gamma(F) = \sum_{k \in \Gamma(j)} \gamma(F_k) + f_j \quad \text{for } j = i.$$

Proof. (i) $j \neq i$. Let $\gamma = \sum_{k \in \Gamma(j)} \gamma(F_k) + w_j d_{ij}$. Then

$$\gamma = \sum_{k \in \Gamma(j)} \left(\sum_{r \mid r \in V_k \text{ and } r \text{ is a root of } F_k} f_r + \sum_{s \in V_k} w_s d_{r^{F_k}(s), s} \right) + w_j d_{i,j}$$

$$= \sum_{k \in \Gamma(j)} \sum_{r \mid r \in V_k \text{ and } r \text{ is a root of } F_k} f_r + \sum_{k \in \Gamma(j)} \sum_{s \in V_k} w_s d_{r^{F_k}(s), s} + w_j d_{i,j}.$$

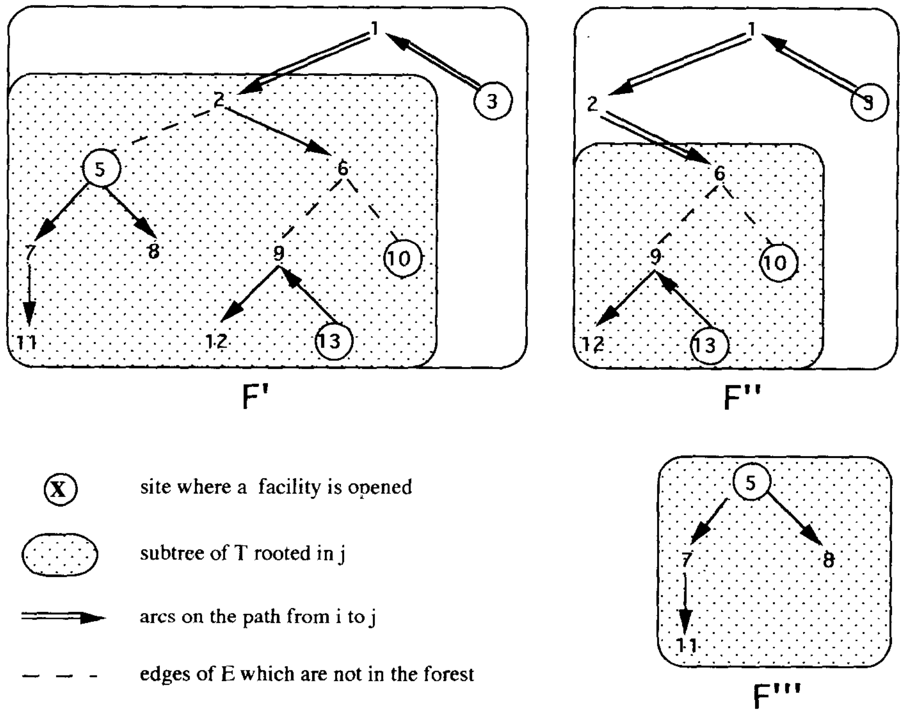

Fig. 2. Examples of forests of $\mathscr{F}(i,j)$. F' is a forest of $\mathscr{F}_{3,2}$; F'' is a forest of $\mathscr{F}_{3,6}$; F''' is a forest of $\mathscr{F}_{5,5}$; $r^{F'}(6) = r^{F'}(2) = 3$; $\Gamma(2) = \{5, 6\}$; $F' = F'' \cup F'''$ (Lemma 1).

The set of roots of F belonging to V_j is the union, for all $k \in \Gamma(j)$, of the set of roots of F_k belonging to V_k. Now, for all $k \in \Gamma(j)$ and all $s \in V_k$, $r^{F_k}(s) = r^F(s)$. Therefore, since $\{\{V_k\}_{k \in \Gamma(j)}, \{j\}\}$ is a partition of V_j, we have

$$\gamma = \sum_{r \mid r \in V_j \text{ and } r \text{ is a root of } F} f_r + \sum_{s \in V_j} w_s d_{r^F(s),s} - w_j d_{r^F(j),j} + w_j d_{i,j}$$

and on noting that $r^F(j) = i$, we deduce, by definition of $\gamma(F)$, $\gamma = \gamma(F)$.

(ii) $j = i$. Let $\gamma = \sum_{k \in \Gamma(j)} \gamma(F_k) + f_j$. The set of roots of F belonging to V_j is the union, for all $k \in \Gamma(j)$, of the set of roots of F_k and $\{j\}$. Now, for all $k \in \Gamma(j)$ and all $s \in V_k$, $r^{F_k}(s) = r^F(s)$. Therefore, since $\{\{V_k\}_{k \in \Gamma(j)}, \{j\}\}$ is a partition of V_j and since $\{j\}$ is a root of F, we have

$$\gamma = \left(\sum_{r \mid r \in V_j \text{ and } r \text{ is a root of } F} f_r - f_j \right) + \left(\sum_{s \in V_j} w_s d_{r^F(s),s} - w_j d_{r^F(j),j} \right) + f_j$$

and on noting that $r^F(j) = j$ and $d_{j,j} = 0$, we deduce, by definition of $\gamma(F)$, $\gamma = \gamma(F)$. □

Now define, for all $(i, j) \in \{1, \ldots, n\}^2$, the quantities c_{ij}:

$$c_{ij} = \begin{cases} \sum_{k \in \Gamma(j)} \min\{c_{rk}\colon r \in V_k \cup \{i\}\} + w_j d_{i,j}, & \text{if } i \notin V_j, \\ \sum_{k \in \Gamma(j)} \min\{c_{rk}\colon r \in V_k \cup \{i\}\} + f_j, & \text{if } i = j, \\ c_{il} + \sum_{k \in \Gamma(j), k \neq l} \min\{c_{rk}\colon r \in V_k \cup \{i\}\} + w_j d_{i,j}, & \text{if } i \in V_l,\ l \in \Gamma(j). \end{cases}$$

Theorem 3. *For all $(i, j) \in \{l, \ldots, n\}^2$*

c_{ij} is the minimal cost of a forest in $\mathscr{F}(i, j)$. $(\mathscr{P})$

Proof. The proof is by recurrence.

We first prove the property for all $(i, j) \in \{1, \ldots, n\}^2$ such that j is a leaf of T $(\Gamma(j) = \emptyset)$:

– if $j = i$, $\mathscr{F}(i, j) = \{F\}$ with $F = \{\{j\}, \emptyset\}$; the cost of F is by definition f_j which is indeed equal to c_{jj}.

– if $j \neq i$, $\mathscr{F}(i, j) = \{F\}$ with $F = (\mathscr{V}(\mu_{ij}), \mathscr{A}(\mu_{ij}))$; the cost of F is by definition $w_j d_{i,j}$ which is indeed equal to c_{ij}.

Now consider an index $j \in \{1, \ldots, n\}$ which is not a leaf of T. Suppose that the property $(\mathscr{P})$ is true for all (r, k), $k \in \Gamma(j)$, $r \in \{1, \ldots, n\}$ and prove that it is true for all (i, j), $i \in \{1, \ldots, n\}$. Fig. 3 gives an illustration of the recurrence.

Let $\gamma^* = \min\{\gamma(F)\colon F \in \mathscr{F}(i, j)\}$ and let F^* be a forest of $\mathscr{F}(i, j)$ such that $\gamma(F^*) = \gamma^*$. From Lemma 1(i), we know that $\forall k \in \Gamma(j)$, $\exists F_k^* \in \mathscr{F}(r^{F^*}(k), k)$ such that $F^* = \bigcup_{k \in \Gamma(j)} F_k^* \cup \mu_{ij}$. Since $r^{F^*}(k) \in V_k \cup \{i\}$, we get from Lemma 2,

$$\gamma^* = \sum_{k \in \Gamma(j)} \gamma(F_k^*) + w_j d_{i,j} \quad \text{if } j \neq i$$

and

$$\gamma^* = \sum_{k \in \Gamma(j)} \gamma(F_k^*) + f_j \quad \text{if } j = i.$$

For all $k \in \Gamma(j)$ such that $i \notin V_k$, we claim that

$$\gamma(F_k^*) = \min\{\min\{\gamma(F_k)\colon F_k \in \mathscr{F}(r, k)\},\ r \in V_k \cup \{i\}\}.$$

Indeed suppose there exists $\hat{r} \in V_{\hat{k}} \cup \{i\}$ and $F'_{\hat{k}} \in \mathscr{F}(\hat{r}, \hat{k})$ such that $\gamma(F'_{\hat{k}}) < \gamma(F^*_{\hat{k}})$ for some $\hat{k} \in \Gamma(j)$; from Lemma 1(ii) the graph F' obtained by replacing $F^*_{\hat{k}}$ by $F'_{\hat{k}}$ in $\bigcup_{k \in \Gamma(j)} F_k^* \cup \mu_{ij}$ is a forest F' of $\mathscr{F}(i, j)$ which, by Lemma 2, has a cost $\gamma(F') < \gamma^*$; this contradicts our assumption.

In the same way if $i \in V_l$, $l \in \Gamma(j)$ then $r^{F^*}(l) = i$, $F_l^* \in \mathscr{F}(i, l)$ and we can conclude that $\gamma(F_l^*) = \min\{\gamma(F_l)\colon F_l \in \mathscr{F}(i, l)\}$.

Then, by the recurrence hypothesis we have,

$$\gamma(F_k^*) = \min\{c_{rk},\ r \in V_k \cup \{i\}\}, \quad \forall k \in \Gamma(j) \text{ such that } i \notin V_k,$$

and

$$\gamma(F_l^*) = c_{il}, \quad \text{if } i \in V_l,\ l \in \Gamma(j).$$

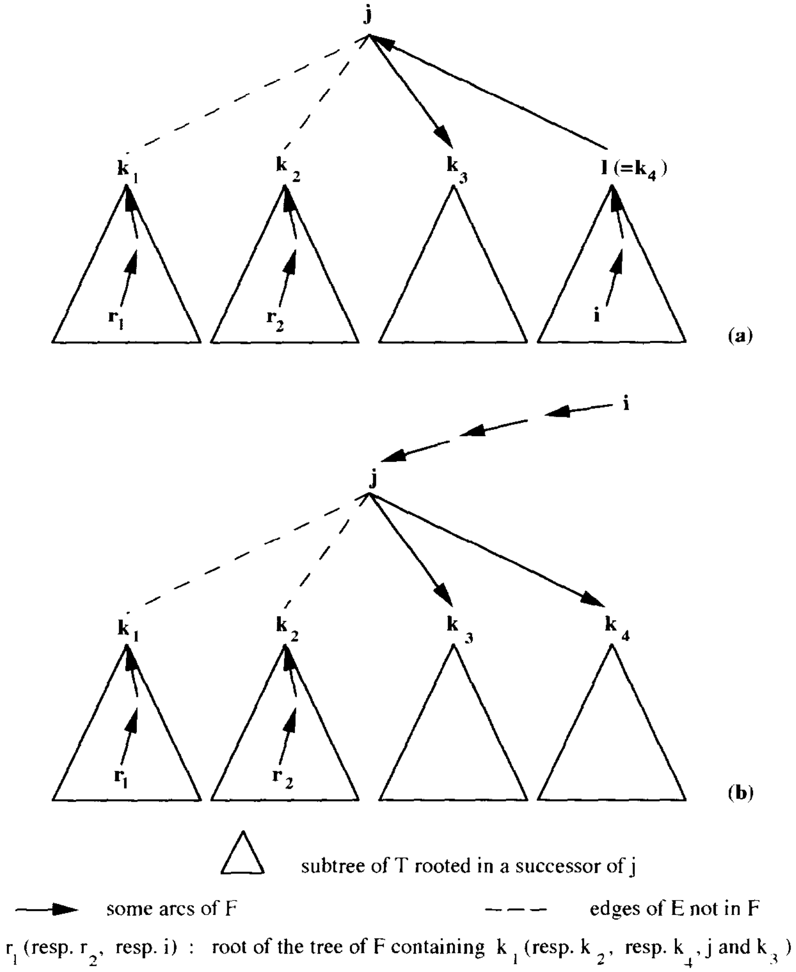

Fig. 3. (a) Structure of a forest F of $\mathscr{F}(i,j)$ when $i \in V_1$, $l \in \Gamma(j)$. (b) Structure of a forest F of $\mathscr{F}(i,j)$ when $i \notin V_j \backslash \{j\}$.

Therefore

$$\gamma^* = \begin{cases} \sum_{k \in \Gamma(j)} \min\{c_{rk}\colon r \in V_k \cup \{i\}\} + w_j d_{i,j} = c_{ij}, & \text{if } i \notin V_j, \\ \sum_{k \in \Gamma(j)} \min\{c_{rk}\colon r \in V_k \cup \{i\}\} + f_j = c_{jj}, & \text{if } i = j, \\ c_{il} + \sum_{k \in \Gamma(j), k \neq l} \min\{c_{rk}\colon r \in V_k \cup \{i\}\} & \\ \quad + w_j d_{i,j} = c_{ij}, & \text{if } i \in V_l, l \in \Gamma(j). \quad \square \end{cases}$$

4. The algorithm

The essential idea of the algorithm is to progressively compute the c_{ij}, beginning with the leaves of T and ending with the root. Indeed, by definition of c_{ij}, it is clear that, for a given j, we can compute c_{ij} for all $i \in \{1,\dots,n\}$ when we know c_{ik} for all (i,k) such that $k \in \Gamma(j)$ and $i \in \{1,\dots,n\}$. If 1 is the root of T, once c_{i1} is determined for all $i \in \{1,\dots,n\}$, an optimal value of the considered u.p.l.p. is obviously equal to $\min\{c_{i1}: i \in \{1,\dots,n\}\}$.

For all $s \in \{1,\dots,n\}$ let $c_s^* = \min\{c_{rs}: r \in V_s\}$. For all $(i,j) \in \{1,\dots,n\}^2$, the c_{ij} can be written

$$c_{ij} = \begin{cases} w_j d_{i,j} + \sum_{k \in \Gamma(j)} \min\{c_k^*, c_{ik}\}, & \text{if } i \notin V_j, \\ f_j + \sum_{k \in \Gamma(j)} \min\{c_k^*, c_{jk}^*\}, & \text{if } i = j, \\ w_j d_{i,j} + c_{il} + \sum_{k \in \Gamma(j), k \neq l} \min\{c_k^*, c_{ik}\}, & \text{if } i \in V_l,\ l \in \Gamma(j). \end{cases}$$

We can now present the algorithm for solving the u.p.l.p. on trees.

Algorithm for solving the uncapacited plant location problem on trees.

begin

1. Transform the tree $T = (V, E)$ into the rooted tree $T = (V, A)$ and let $1, 2, \dots, n$ be a breadth-first ordering of the vertices of V (see Fig. 1(b))
2. **for** $j = n, n-1, n-2, \dots, 1$ **do**
3. **for all** $i \in V \setminus V_j$ **do**
   $c_{ij} \leftarrow w_j d_{i,j} + \sum_{k \in \Gamma(j)} \min\{c_k^*, c_{ik}\}$
   end do
4. $c_{jj} \leftarrow f_j + \sum_{k \in \Gamma(j)} \min\{c_k^*, c_{jk}\}$; $c_j^* \leftarrow c_{jj}$
5. **for all** $l \in \Gamma(j)$ **do**
   for all $i \in V_l$ **do**
   $c_{ij} \leftarrow w_j d_{i,j} + c_{il} + \sum_{k \in \Gamma(j), k \neq l} \min\{c_k^*, c_{ik}\}$
   if $c_{ij} < c_j^*$ **then** $c_j^* \leftarrow c_{ij}$ **endif**
   end do
   end do
   end do

"*comment*: The value of an optimal solution is given by c_j^* for the last j examined during the loop 2, i.e., by c_1^*, 1 being the root of $T = (V, A)$"

end

4. Run time analysis

Theorem 4. *The previous algorithm solves the uncapacited plant location problem on trees in* $O(n^2)$ *time.*

Proof. The time required to execute step 1 is $O(n)$. Now consider loop 3, statement 4 and loop 5. For a given vertex j we compute the c_{ij} for all $i \in \{1,\dots,n\}$ and each computation of a c_{ij} requires $O(|\Gamma(j)|)$ time. Therefore the total time required by loop 3, statement 4 and loop 5, for a given j, is $O(n \cdot |\Gamma(j)|)$. Thus the total running time of the for loop 2 is $O(\sum_{j \in V} n \cdot |\Gamma(j)|) = O(n \sum_{j \in V} |\Gamma(j)|)$ and finally, since $\sum_{j \in V} |\Gamma(j)| = n - 1$ we have $O(n^2)$ as total running time of the algorithm. □

Acknowledgement

We thank the referees for their constructive remarks.

References

[1] A.J. Hoffman, A. Kolen and M. Sakarovitch, Totally-balanced and greedy matrices, SIAM J. Algebraic Discrete Methods 6 (1985) 721–730.
[2] A. Kolen, Solving covering problems and the uncapacited plant location problem on trees, European J. Oper. Res. 12 (1983) 266–278.
[3] J. Krarup and P.M. Pruzan, The simple plant location problem, European J. Oper. Res. 12 (1983) 36–81.
[4] M. Schaffers, A polynomial algorithm for the single source network flow design problem on series-parallel graphs, CORE Discussion Paper 9062 CORE, Université Catholique de Louvain, Louvain-la-neuve (1990).[1]

[1] This reference (pointed out by one of the referees) contains an algorithm for a more general problem which seems possible to specialize to obtain the same complexity result for the u.p.l.p. on trees.